Laboratorium 1

Łuszczek Patryk
272707

6 maja 2025

1 Cel zadania

Celem zadania było stworzenie aplikacji typu PWA - Progressive Web App oraz wdrożenie jej w system chmurowy. Progressive Web App to aplikacja internetowa, która wykorzystuje możliwości nowoczesnych przeglądarek internetowych, aby zapewnić użytkownikom doświadczenie podobne do aplikacji natywnych. Aplikacje PWA są responsywne, szybkie i mogą działać offline, co czyni je idealnym rozwiązaniem dla użytkowników mobilnych.

2 Opis aplikaci

Aplikacja służy do proponowania przepisow kulinarnych pobieranych z API dostępnego pod adresem `https://www.themealdb.com`. Użytkownik po wejściu do aplikacji może przeglądać losowo pobierane przepisy oraz zapisywać je do ulubionych. Przepisy zapisane do ulubionych są następnie przechowywane w pamięci, co pozwala na ich późniejsze przeglądanie nawet w trybie offline.

3 Opis procesu

3.1 Tworzenie aplikacji

W celu umożliwienia dzialaności aplikacji jako PWA został utworzony "service worker", który obsługuje cache'owanie zasobów aplikacji oraz zapewnienie możliwości działania offline. W czasie instalacji aplikacji SW zapisuje do pamięci podręcznej zdefiniowane pliki, w tym przypadku były to pliki `index.html`, `style.css`, `main.js`. Równie ważnym krokiem było utworzenie pliku manifestu, który definiuje podstawowe informacji aplikacji takie jak nazwa, ikona czy kolor motywu.

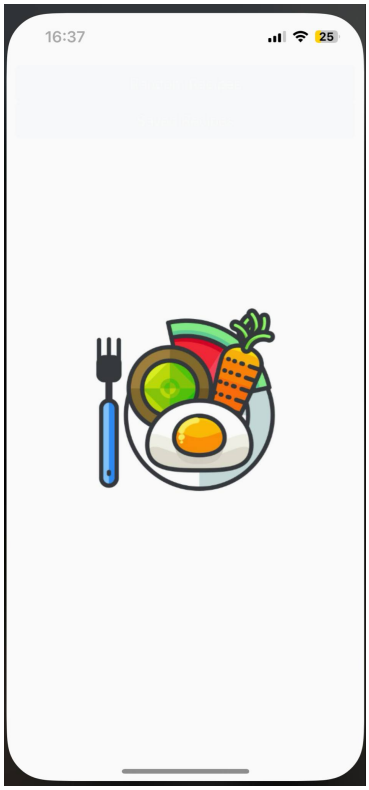
```
const cacheName = "piac-pwa-v1";
const baseUrl = "/pwa-js/"
const filesToCache = [
  baseUrl + "index.html",
  baseUrl + "manifest.json",
  baseUrl + "images/icons/icon-192x192.png",
  baseUrl + "styles.css",
  baseUrl + "js/main.js"
]

self.addEventListener("install", (event) => {
  event.waitUntil(
    caches.open(cacheName).then((cache) => {
      return cache.addAll(filesToCache).catch(err => {
        console.error("Caching failed:", err);
      });
    })
  );
});

self.addEventListener("fetch", (event) => {
  event.respondWith(
    caches.match(event.request).then((response) => {
      if (response) {
        return response;
      }      You, 4 weeks ago • added files …
      return fetch(event.request).then((fetchResponse) => {
        if (event.request.method === "GET") {
          return caches.open(cacheName).then((cache) => {
            cache.put(event.request, fetchResponse.clone());
            return fetchResponse;
          });
```

Rysunek 1: Service worker

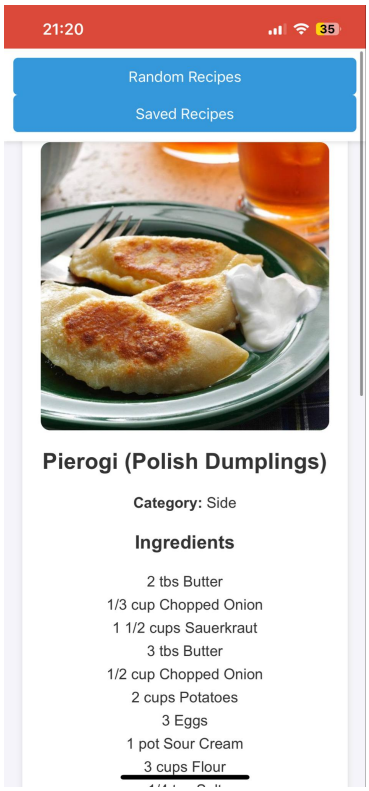
Dodatkowo w pliku `index.html` dodano odpowiednie tagi linkujące do pliku manifestu oraz zawierające ikony aplikacji wyświetlane na urządzeniach mobilnych.

3.2 Wygląd, funkcjonalność i instalacja aplikacji

Jak już wcześniej wspomniano, aplikacja służy do przeglądania nowych przepisów kulinarnych. Po wejściu do aplikacji pokazuje się splash screen, a następnie przedstawiany jest losowy przepis (o ile jest dostęp do internetu).



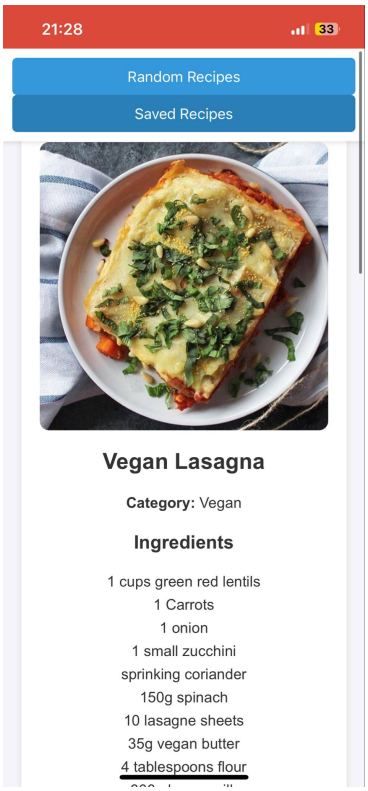
Rysunek 2: Splash screen



Rysunek 3: Losowy przepis

Jeśli nie ma dostępu do internetu, aplikacja umożliwia tylko przeglądanie ulubionych

przepisów.



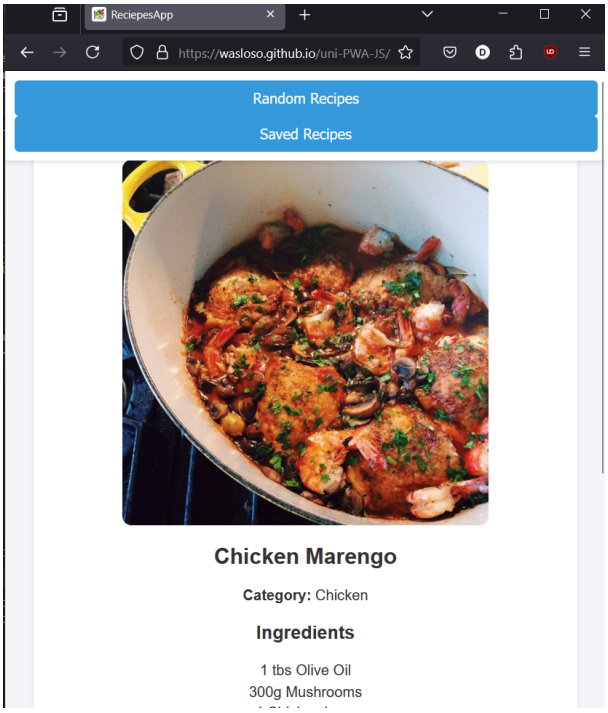
Rysunek 4: Losowy przepis

W przypadku urządzeń z systemem iOS istnieje łatwy sposób "zainstalowania" takiej aplikacji, co też zostało przetestowane, a wszystkie screeny pochodza właśnie z tak pobranej aplikacji.

Rysunek 5: Aplikacja pobrana na iOS

4 Wdrożenie aplikacji

Aplikacja została wdrożona na platformie GitHub Pages. Nie sprawiało to żadnych problemów, oprócz potrzeby zmian linków, aby odwzorowały poprawną lokalizację plików.



Rysunek 6: Aplikacja wdrożona na GitHub Pages